Fast recursive grayscale morphology operators: from the algorithm to the pipeline architecture

Olivier Déforges, Nicolas Normand, Marie Babel

2 **Journal of Real Time Image Processing, manuscript No.**
(will be inserted by the editor)

O. Déforges · N. Normand · M. Babel.

Fast Recursive Grayscale Morphology Operators: from the Algorithm to the Pipeline Architecture

the date of receipt and acceptance should be inserted later

Abstract This paper presents a new algorithm for an efficient computation of morphological operations for gray images and its specific hardware. The method is based on a new recursive morphological decomposition method of 8-convex structuring elements by only causal two-pixel structuring elements (2PSE). Whatever the element size, erosion or/and dilation can then be performed during a unique raster-like image scan involving a fixed reduced analysis neighborhood. The resulting process offers low computation complexity combined with easy description of the element form. The dedicated hardware is generic and fully regular, built from elementary interconnected stages. It has been synthesized into an FPGA and achieves high frequency performances for any shape and size of structuring element.

Keywords: Mathematical morphology, 8-convex structuring element operators, regular dedicated pipeline architecture

1 Introduction

Mathematical morphology was first introduced as a method to measure binary objects, but soon became a complete theory based on set operations [Serra (1982)]. Morphology relies on the use of set operators (intersection, union, inclusion, complement) to transform an image. The transformed image usually has fewer details, but its main characteristics are still present. Once the image has been simplified, measurements can be computed to give a quantitative analysis of the image. Morphological transformation is based on a structuring element ($\boldsymbol{B}$) characterized by its shape, size and centre location, also called origin. Each pixel in an image is compared with $\boldsymbol{B}$ by moving $\boldsymbol{B}$ so that its center hits the pixel. Depending on the type of morphological transformation, the pixel value is set to the minimal or maximal value of the pixels in the translated structuring element.

Performing a morphological operation directly from its formal definition would imply that all the neighbors present within the structuring element should be sought for each pixel. Of course, this process becomes time consuming for large elements. Fortunately, mathematical morphology properties led to the introduction of methods which partially solve this problem.

The main goal of the decomposition algorithms is to reduce the number of basic dilations needed to generate the desired structuring element. Different ways to decompose structuring elements have been investigated exploiting two fundamental properties: Minkowski additions [Xu (1991), Chen and Haralick (1995)] and set unions of elements [Wang and Bertrand (1988), Ji et al (1989)]. More particularly, Chen has defined a decomposition method using only two non-connected pixel elements. On the other hand, Xu demonstrated how any 8-convex polygon could be constructed from a set of elements included in a 4-neighborhood.

Mathematical morphology is a tool commonly used in embedded vision systems, so a great deal of work has been done to design hardware implementation. The quality of an architecture dedicated to mathematical morphology is generally measured by the ability to describe different element shapes versus hardware efficiency e.g. memory and computation requirements, or number of image scans. Of course, decomposition element methods can help the design of efficient dedicated hardware by limiting the number of operations.

Chien et al (2005) proposed a comparative table among existing architectures for a dilation by a 7×7

O. Déforges · M. Babel
UMR CNRS 6164 IETR, INSA Rennes, 20 av. des Buttes de Coesmes, 35043 Rennes, FRANCE
Tel.: +33-2-23238286
Fax: +33-2-23238262
E-mail: {odeforge, mabel}@insa-rennes.fr

N. Normand
UMR CNRS 6597 IRCCYN, Ecole Polytechnique Nantes, La Chantrerie, BP 60601, 44306 Nantes, FRANCE
Tel.: +33-2-40683044
Fax: +33-2-40683232
E-mail: nnormand@polytech.univ-nantes.fr

Architecture	Support non flat structuring element	Support non rectangular structuring element	Comparator counts	Number of delay elements	Cycles required per frames
Pitas	No	No	8	7L + 7	L(H+7)+6
Coltuc	No	No	6	6L +6	L(H+6)+5
Ong	Yes	No	7	[6L +6]+1	7LH
Diamantaras	Yes	Yes	48	[6L +42]	L(H+6)
Ruetz	No	No	12	6L +6	L(H+6)+5
Sheu	Yes	No	13	6L + 20	7LH + 8
Chien	No	Partially	6	6L+6	L(H+6)+5
This work	No	Yes	12	6L+6	L(H+6)+5

Table 1 Flexibility and implementation efficiency comparison between state-of-the-art architectures for a dilation by a 7×7 square structuring element

structuring element (see Table 1). Chien architecture relies on the principle of PRR (Partial-Result-Reuse), trying to avoid redundant computations. This concept has been used earlier by Pitas (1989) and Ong and Sunwoo (1997), but their architectures were not fully optimized. Coltuc and Pitas (1998) proposed an optimal solution but only for rectangular elements, and Chien architecture is an extension of some other shapes. A comparison between PRR and the proposed method will be later discussed.

Diamantaras and Kung (1997) architecture provides the best flexibility, supporting both non flat and arbitrarily-shaped elements. However, it is at the expense of complexity. The architecture is built as a systolic array, and one Processor Element is defined for each pixel of the element. Ruetz and Brodersen (1987), Sheu et al (1992) both proposed pipelined implementations deduced from decomposition element techniques, that are also limited to flat rectangular elements. The performance and capability of their approach will be further discussed in the section on "Results".

Other works concern computational efficiency for grayscale morphological operators. Van Herk (1992) proposed a very efficient method for grayscale erosions and dilations, requiring only three comparisons for 1D structuring elements, whatever the element size. It relies on a recursive forward and backward ranking of pixel values inside temporary buffers. Extension to 2D rectangular structuring elements is achieved by consecutively applying the same process to rows and columns, for a total of 6 comparisons. 2D circular symmetric filtering can also be considered by running the process along the two diagonal directions, doubling the number of operations. Soille et al (1996) introduced an extension of this algorithm to any angle 1D structuring elements. Gil and Kimmel (2002) further improved the initial algorithm to reach 1.5 comparisons for 1D erosions and dilations, thanks to a sliding window technique. The major drawback is that this algorithm induces non regular data flow processing, as well as the generation of ordered lists, which are not compatible with efficient implementations. The other drawback is that all Van Herk-based techniques have been designed and optimized in 1D space: 2D extension requires a second pass, and is mainly restricted to rectangular elements shapes. The technique proposed in this paper performs one pass erosion or dilation, considering more generalized 2D convex shapes.

In this paper, we address the problem of grayscale morphology with a restricted class of discrete convex polygons (2). We then introduce a fast recursive technique for grayscale dilations, relying on this particular decomposition scheme. The process regularity also enables the design of a pipeline architecture (section 3). We summarize the results in section 4.

2 Fast recursive morphological operators

2.1 Grayscale morphology

Applying morphological operations on an image involve basic templates called structuring elements, used as shape parameters for the operations. The two most fundamental morphological operations are dilation and erosion. Grayscale morphology is a natural extension of binary morphology, using the Umbra and Top operators. In most of the cases grayscale morphology is generally restricted to plane structuring elements leading to simple dilation and erosion operations based on Min and Max functions.

A gray image A is a function whose domain is a subset of the two dimensional digitized space $Z \times Z$. For any point $p \in Z \times Z$, the translation of A by p is defined by

$$(A)_p = \{a + p \mid \forall a \in A\} \tag{1}$$

and the erosion and dilation by the structuring element B are respectively given by

$$\begin{aligned} A \ominus B &= min\left\{(A)_p | p \in B\right\} = \min_{p \in B} (A)_p \,, \\ A \oplus B &= max\left\{(A)_p | p \in B\right\} = \max_{p \in B} (A)_p \,. \end{aligned} \tag{2}$$

Some useful properties must be presented to justify the algorithm proposed later.

Translation. The dilation by the translated of a structuring element can be easier to compute than the dilation by the element itself. The following properties demonstrate the equivalence of operations (except for the translation):

$$\begin{aligned} A \ominus (B)_p &= (A)_p \ominus B = (A \ominus B)_p \\ A \oplus (B)_p &= (A)_p \oplus B = (A \oplus B)_p \end{aligned} \tag{3}$$

Associativity. Dilations (respectively erosions) by a series of structuring elements are equivalent to a dilation (resp. erosion) by the Minkowski addition of all the elements of the series:

$$\begin{aligned} (A \ominus B) \ominus C &= A \ominus (B \oplus C) \\ (A \oplus B) \oplus C &= A \oplus (B \oplus C) \end{aligned} \tag{4}$$

This property justifies a lot of algorithms since, as long as a structuring element can be decomposed by Minkowski additions of smaller elements, it is generally advantageous from a computational point of view to dilate or erode the image successively by all these elements. This property is often used conjointly with the previous one. Hence dilating by a series of translated elements is equivalent to the combination of one global translation and dilations by the elements themselves. The number of operations per pixel can then be drastically reduced, but the major drawback is that the image has to be scanned several times.

Union of structuring elements. The other way to split a structuring element is to realize a set union of smaller elements. The dilation (resp. erosion) by a set union of structuring elements is then equal to the union of the dilations (resp. intersection of the erosions) by all the structuring elements:

$$\begin{aligned} A \ominus (B \cup C) &= (A \ominus B) \cap (A \ominus C) \\ A \oplus (B \cup C) &= (A \oplus B) \cup (A \oplus C) \end{aligned} \tag{5}$$

Opening/closing operators. Erosion and dilation operations are frequently combined to provide powerful filters. An erosion (resp. dilation) by B followed by a dilation (resp. erosion) by $\tilde{B}$, the symmetrical element of B, is called an opening (resp. closing) operation:

$$\begin{aligned} &Opening : A \circ B = (A \ominus B) \oplus \tilde{B} \\ &Closing : A \bullet B = (A \oplus B) \ominus \tilde{B} \end{aligned} \tag{6}$$

2.2 H-convex polygons

A shape in $R \times R$ is said to be H-convex if it corresponds to the (possibly unbounded) intersection of a set of half-planes [Boltyanskii and Soltan (1978)], or, equivalently, if it contains all line segments defined by a couple of its points. A discrete set of points (in $Z \times Z$) is said to be H-convex if it is equal to the intersection of a H-convex shape of $R \times R$ with $Z \times Z$ [Eckhardt (2001)]. As a consequence, a discrete H-convex shape is not necessarily connected in the usual sense (4- or 8-connexity). A bounded discrete H-convex shape is by definition the convex hull of a finite set of discrete points, hence is a polygon. In [Normand (2003)], a method to decompose arbitrary discrete H-convex polygons is presented and applied to the construction of structuring elements for binary morphology.

We call 8-convex polygon, a 8-connected discrete H-convex polygon whose edge directions are restricted to 0°, 45°, 90° and 135° like the structuring elements used in [Xu (1991)]. In the following, we show that 8-convex polygons can be decomposed with a fixed (4) number of 2PSEs whereas this amount varies for general H-convex polygons.

2.3 Xu decomposition

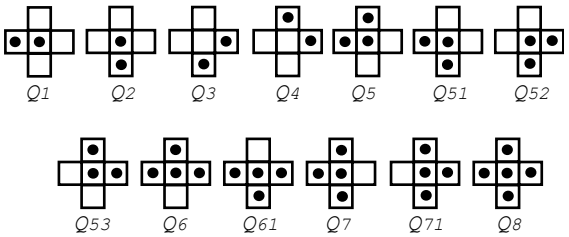

Fig. 1 Generating set of structuring elements for 8-convex polygons

Xu proved that any 8-convex polygon can be built by dilations with a restricted set of generating structuring elements [Xu (1991)]. The 13 generating elements proposed by Xu are shown in figure 1. They constitute the minimal set to generate any 8-convex polygon, i.e. any convex polygon whose border is 8-connected. Their origin is normally the center of the cross but in order to restrict later considerations to causal relationships, we replace the elements by their translated version whose origin is the last point in a raster scan order. According to (3) it introduces only a translation of the resulting image. In the following the translated version of an element Q_x will be referred to Q'_x. Figure 2 provides an example of a 8-convex structuring element and its Xu decomposition.

2.4 Causal Recursive Decomposition of structuring elements

We have demonstrated in [Normand (2003)] that all the Xu's decomposition elements can themselves be built as

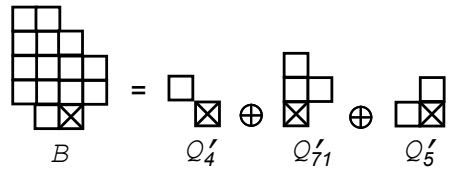

Fig. 2 Example of structuring element decomposition

unions and dilations of the four Two-Pixel Structuring Elements (2PSEs) depicted in figure 3.a. By extension, it implies that these 2PSEs are also sufficient to build any 8-convex element as a series of both set unions and dilations. The main advantage of using 2PSEs is that they involve a restricted 8-connected causal neighborhood of analysis defined by four elementary translations $p_k, k \in \{1 \dots 4\}$ (see figure 3.b).

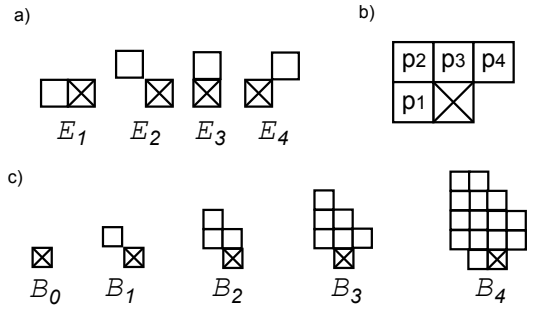

Fig. 3 2PSE decomposition (a) Elementary elements, (b) corresponding translations and (c) example of growing structuring family

We have established in [Normand (2003)] that any convex structuring element can be iteratively constructed: for each step of the construction, already-built elements can be joined together in any combination of translations and set unions. We have also introduced a deterministic method for convex shape decomposition.

Restricting elementary structuring elements to the set of four 2PSEs depicted in figure 3.b not only leads to consideration of only 8-convex elements but also limits the analysis space to an 8-connected causal neighborhood. In this case, each 8-convex B_i can be recursively constructed according to:

$$B_{-1} = \emptyset; B_0 = \{O\}; \forall i > 1, B_i = \bigcup_{k \in \{1..4\}} B_{I_k(i)} \oplus E_k, \quad (7)$$

where $I_k(i)$ ($I_k(i) < i$) denotes which previous element belonging to the family and dilated by E_k is used to build B_i. The empty set is represented by $I_k(i) = -1$ and B_0 only contains the origin. $\{B_i\}_{i=0..N}$ forms an increasing family of structuring elements as long as $B_{i-1} \subset B_i$.

Going on with the previous example we see that the translated Xu elements (Q'_4, Q'_{71}, and Q'_5) can be built by dilations and unions of 2PSEs:

$$Q'_4 = E_2,\ Q'_{71} = (E_3 \oplus E_3) \cup E_4,\ Q'_5 = E_1 \cup E_3 \quad (8)$$

In most of the cases (9/13), the Q' are expressed by an unique set of unions, like Q'_4 and Q'_5, and the recursive relationship from (7) is straightforward. To change over from the Xu decomposition scheme to the proposed one for the four left structuring elements, we introduce an intermediate step: let us assume that $B_i = B_{i-2} \oplus Q'_{71}$, according to (5) and (8)we get

$$\begin{aligned} B_i &= (B_{i-2} \oplus E_3 \oplus E_3) \cup (B_{i-2} \oplus E_4) \\ &= (B_{i-1} \oplus E_3) \cup (B_{i-2} \oplus E_4) \end{aligned} \quad (9)$$

where $B_{i-1} = B_{i-2} \oplus E_3$. From relation (7) it gives $I_3(i-1) = i-2$, $I_3(i) = i-1$ and $I_4(i) = i-2$. The corresponding growing family is given below, and the iterative construction is depicted in figure 3.c.

$$\begin{array}{ll} Dil.\ by\ Q'_4 & B_1 = B_0 \oplus E_2 \\ Dil.\ by\ Q'_{71}\ (step\ 1) & B_2 = B_1 \oplus E_3 \\ Dil.\ by\ Q'_{71}\ (step\ 2) & B_3 = (B_2 \oplus E_3) \cup (B_1 \oplus E_4) \\ Dil.\ by\ Q'_5 & B_4 = (B_3 \oplus E_1) \cup (B_3 \oplus E_3) \end{array} \quad (10)$$

A growing structuring element family is then very easily described by providing only the four $I_k(i)$ indices per level. In practice most of the structuring elements used are symmetrical ones. In that particular case, all of them (except crosses) can be obtained when restricting formulae (7) to only elementary dilations. The simplified associated expression is given by:

$$B_i = B_{i-1} \oplus E_i,\ E_i \in \{E_1, E_2, E_3, E_4\} \quad (11)$$

Only one index per level is necessary here for the decomposition description. The alternatively use of E_1, E_2, E_3, E_4 leads to a circular structuring element, while limiting the construction to E_1 and E_3 generates rectangular ones.

2.5 Grayscale image recursive dilation and erosion

In [Normand (2003)], we took advantage of the previous structuring element decomposition method by introducing a fast binary erosion operator. The binary case, which presents a fixed number of operations whatever the structuring element, cannot be used to expand the grayscale morphology. Even if comparable performances cannot be achieved, we propose here a fast algorithm for grayscale erosion and dilation operators. The principle will be detailed only for dilation.

First, consider that as the four 2PSEs contain the origin, an elementary dilation by E_k is equivalent to the union of the element and the element translated by p_k:

$$B \oplus E_k = B \cup (B)_{p_k} \quad (12)$$

Let $\{B_i\}_{i=0..N}$ be a family of growing structuring elements constructed recursively by union sets and elementary dilations according to (7). In the following D_{B_i} states the dilation of A by B_i. Based on the 2PSE decomposition a recursive computation of D_{B_i} is possible:

$$\begin{aligned}
&D_{B_i} = A \oplus B_i \\
&= A \oplus \left(\bigcup_{k\in\{1..4\}} B_{I_k(i)} \oplus E_k \right) \\
&= A \oplus \left(\bigcup_{k\in\{1..4\}} B_{I_k(i)} \cup \left(B_{I_k(i)}\right)_{p_k} \right) \\
&= Max \left\{ (A)_p \,|p \in \bigcup_{k\in\{1..4\}} B_{I_k(i)} \cup \left(B_{I_k(i)}\right)_{p_k} \right\} \\
&= \underset{k\in\{1..4\}}{Max} \left(Max \left\{ (A)_p \,|p \in \left(B_{I_k(i)} \cup \left(B_{I_k(i)}\right)_{p_k} \right) \right\} \right) \\
&= \underset{k\in\{1..4\}}{Max} \left(Max \left\{ (A)_p \,|p \in B_{I_k(i)} \right\}, \right. \\
&\qquad \left. Max \left\{ (A)_p \,|p \in \left(B_{I_k(i)}\right)_{p_k} \right\} \right) \\
&= \underset{k\in\{1..4\}}{Max} \left(D_{B_{I_k(i)}}, \left(D_{B_{I_k(i)}}\right)_{p_k} \right)
\end{aligned} \tag{13}$$

These results show that the dilated values when considering successively all the elements in the family, are recursively obtained by performing a maximum extraction between the values of the dilation by the elements involved to build the current one, at the current position and at positions p_k in the 8-connected causal neighborhood. The process is fully regular because it uses a raster scan of the image.

In this scheme, resources requirements are linear.

- **Complexity** : Processing requirements concern *Max* operators. If N_D and N_U are respectively the total number of elementary dilations and unions used to build B_i, then the algorithm requires only $N_D + N_U$ comparisons to perform the dilation of an image by B_i whatever the shape of the structuring element.
- **Memory** : Only intermediate results $\{D_{B_i}\}_{i=0..N}$ have to be recorded, so the number of stored elements is given by the number of decomposition levels. When only p_1 is involved for one level, the intermediate value corresponds to the previous computed one, and an unique variable is necessary to store the result. In other cases, a line-memory per level is necessary to store intermediate results from the previous line.

Figure 4 shows the progressive steps to get dilated values up to the structuring element B_3 from figure 3. D_{B_3} computation requires five comparisons, and D_{B_4} eight.

When B is a structuring element composed of only dilations (eq. (11) for symmetrical elements), the operations at each stage are reduced to an unique comparison between the dilated values by the previous element at the current position and at position p_i, $p_i \in \{p_1, p_2, p_3, p_4\}$, so that

$$\begin{aligned}
&B_i = B_{i-1} \cup (B_{i-1})_{p_i} \\
&\Rightarrow D_{B_i} = Max \left(D_{B_{i-1}}, \left(D_{B_{i-1}}\right)_{p_i} \right)
\end{aligned} \tag{14}$$

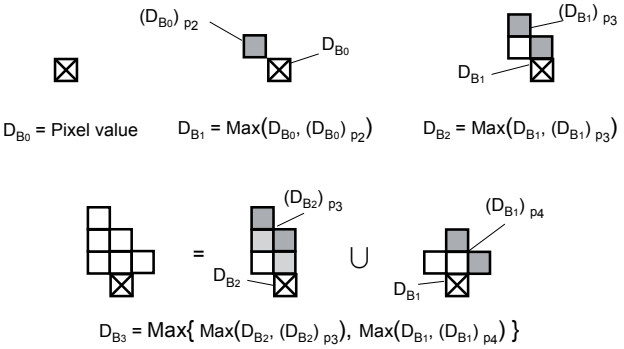

Fig. 4 Recursive dilation process

In this particular case, the number of operations is also given by the number of decomposition levels.

Image borders. To handle the problem of image borders, one has generally to expand the image or to introduce conditional processing. For a dilation operation, a suitable way consists of considering values as 0 outside borders. This can be done here without preliminary image expansion as explained below.

- If L is the image width, then the line-memories have to be of size $(L+2)$.
- Memory-lines have to be initialized to 0 once at the beginning of the process, variables to store results in p_1 position have to be initialized at the beginning of each line process.
- The position values for pixels in the image are considered in the range $[1..L]$. Thus, for pixel position 1 (resp. L) and for p_2 (resp. p_4) neighborhood, line-memory is accessed at address 0 (resp. $L+1$) and provides 0 as a dilated value.

Considering 0 values outside the borders has of course an impact on results near these borders. Assuming that that $D_{B_i} = 0$ if B_i is totally outside the image, it modifies equation (13):

$$D_{B_i} = \underset{k\in\{1..4\}}{Max} \begin{pmatrix} D_{B_{I_k(i)}}, \left(D_{B_{I_k(i)}}\right)_{p_k} \\ if B_{I_k(i)} \subset A \\ D_{B_{I_k(i)}} \quad otherwise. \end{pmatrix} \tag{15}$$

From the previous formulae, one can see that the process automatically adapts the shape of the structuring element near borders and that the actual structuring element expression is:

$$\begin{aligned}
&B_i = \bigcup_{k\in\{1..4\}} B_{I'_k(i)} \cup \left(B_{I''_k(i)}\right)_{p_k}, \; with \\
&I'_k(i) = argmax \left(j \in \{0 \ldots I_k(i)\} \,|B_j \subset A\right) \\
&I''_k(i) = argmax \left(j \in \{0 \ldots I_k(i)\} \,|\, (B_j)_{p_k} \subset A\right)
\end{aligned} \tag{16}$$

To sum up for this part, the proposed method enables an efficient software implementation of dilation and erosion operators with the following main features:

– an unique scan of the image whatever the shape and size of the structuring element,
– an easy and systematic way to describe any 8-convex structuring element,
– limited resources requirements in terms of both memory and computation,
– automatic adaptation to borders without image expansion nor conditional processing.

Further improvements could be made, in particular concerning the number of comparisons. In Chien et al (2005), Chien introduced the concept of Partial-Result-Reuse (PRR). The idea is to limit the number of operations by avoiding redundant comparisons as much as possible and by exploiting already computed data. The example of a 1D structuring element of 8 pixels (called B_4) given by Chien is illustrated figure 5. B_4 is recursively constructed by gathering non-overlapped structuring elements. Intermediate results have to be stored at different distances from the current position : 1 for B_0, 2 for B_1 and 4 for B_3, giving a total of 7 recorded values.

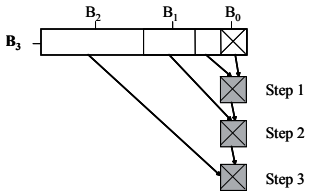

Fig. 5 PRR principle for a 1D structuring element of 8 pixels

If we compare this with our approach, the same structuring element would be built according to :

$$\begin{array}{l} B_7 = B_6 \cup (B_6)_{p_1} \\ \dots \\ B_1 = B_0 \cup (B_0)_{p_1} \end{array} \quad (17)$$

The dilation is performed here after 7 comparisons. Actually, 1D dilations are the worst case for our method in terms of size of the element versus number of comparisons. Concerning memory, both methods require the storage of 7 values but our method involves an unique distance (equal to 1) whereas the depth for PRR is dependant on the element decomposition.

The PRR concept has been extended to square and circular elements. Square representation is the simplest shape leading to the greatest computational reduction. For circular elements (13 pixel elements), Chien's heuristic method requires 6 computation steps, whereas the proposed method requires 8. However, the authors do not propose a systematic method of decomposition. For a dichotomic approach, the problem can be reformulated by trying to find the minimum element B_j of the family (where $j \in [0 \dots i-1]$) such that B_i can be expressed as

$$\begin{array}{l} B_i = B_j \cup (B_j)_{T \times p_i} \\ T \in \{1 \dots \lfloor \frac{i}{2} \rfloor\}, \quad p_i \in \{p_1, p_2, p_3, p_4\} \end{array} \quad (18)$$

The 1D case is simple. Formula (17) can first be expanded, and, by factoring process, it is easy to find that there are four possible solutions meeting the condition set out in (17). Among them, B_3 is the minimum element.

$$\begin{array}{ll} B_7 = B_6 \cup (B_6)_{p_1} & B_7 = B_5 \cup (B_5)_{2p_1} \\ B_7 = B_4 \cup (B_4)_{3p_1} & B_7 = B_3 \cup (B_3)_{4p_1} \end{array} \quad (19)$$

The approach can be extended to 2D space but only for simple element shapes. For more complex elements, PRR is more difficult to determine, and as mentioned before, the benefit of PRR decreases when the complexity of the pattern increases.

Opening and closing Opening and closing operations imply the need to consider symmetrical elements. If B is causal, $\tilde{B}$ is necessarily anti-causal, which would lead to an inverse scan of the image in order to preserve the sequential feature of the process. A better solution takes advantage of the property (3) by defining $B_T = (\tilde{B})_{p_B}$, with p_B a fixed vector such that B_T is causal. Then the use of B_T instead of $\tilde{B}$ simply requires a global image translation of p_B once processing has been completed. For a regular data flow it consists merely of introducing a constant delay in the output flow.

When B is symmetrical, deduced from (11), then B and $\tilde{B}$ have the same shape, and the same structuring element can be used.

3 Specific pipeline architecture

The regularity of the algorithm and the restriction of the relationship area to the 8-connected causal neighborhood enable the design of a fast generic and regular specific architecture, requiring few resources in terms of memory and logic. A dilation implementation suited for an FPGA target is presented in the following paragraphs.

The whole architecture is synchronized by a global clock with a data flow rate of one input and output pixel data for each clock pulse. Each elementary stage computing D_{B_i} from previous values, requires two types of components: a memory module to store and to provide previous results and a combinatory operative part to realize the maximum extraction.

For the following, let T be the clock period, $D_{B_i}(n)$ be the dilation value by the structuring element B_i at time $n \times T$ (n^{th} pixel), L be the size of an image row and M be the number of bits to encode a value.

3.1 Memory module

For easier systematic design of the architecture, a generic memory module has been designed. The knowledge of a previous result in the $\{p_2, p_3, p_4\}$ neighborhood implies

the storage of one row at each step of the decomposition. We have chosen a "dual port RAM" memory solution allowing both a reading and a writing access during one clock cycle. The following temporal relationships avoid the need for multiple address management process in each of the various stages:

$$\begin{aligned}
&\left(D_{B_i}(n)\right))_{p_1} = D_{B_i}(n-1)\\
&\left(D_{B_i}(n)\right)_{p_4} = D_{B_i}(n-(L-1))\\
&\left(D_{B_i}(n)\right)_{p_3} = D_{B_i}(n-L) = \left(D_{B_i}(n-1)\right)_{p_4}\\
&\left(D_{B_i}(n)\right)_{p_2} = D_{B_i}(n-(L+1)) = \left(D_{B_i}(n-1)\right)_{p_3}
\end{aligned} \tag{20}$$

Figure 6 presents the general solution for the memory module. For this example we have considered $L = 7$ (3 bits to encode the address) and M=4. The storage $D_{B_i}(n-1)$ requires only a simple register. The memory is addressed at a fixed relative position $((n+2)\ Modulo\ L)$ to provide $\left(D_{B_i}\right)_{p_4}$ at the next clock pulse. Then the dilated values at the two other positions are available by a temporal pipeline through two registers. The reading and writing addresses have a constant difference (equal to 3) which can be also seen as a temporal delay in terms of clock cycle. Therefore the simplest architectural solution consists of the implementation of three registers.

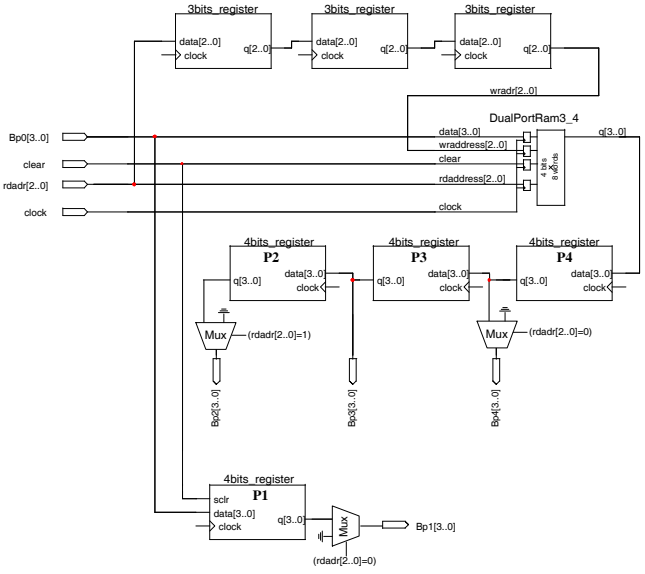

Fig. 6 Generic elementary memory module

Of course all the resources displayed in the memory module are not always necessary. For instance, when only p_1 among the neighborhood is involved it requires only one register. However, these kinds of optimizations are generally automatically performed by synthesis tools removing all unused resources.

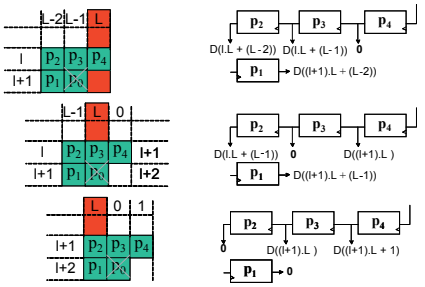

Fig. 7 Border processing

3.2 Image borders

As mentioned before, image borders induce specific processing which can break the data flow regularity of the architecture. To avoid such problem, a solution has been designed. Only one extra storage element is required for the memory-lines, and as borders for both the p_2 and p_4 neighborhoods. Figure 7 illustrates the behavior of the output memory module on right and left borders. To flush the pipeline at the end of each row, and to load values at the beginning, an additional clock pulse is performed. Then a control signal $(wren)$ is generated to avoid to overwrite stored value (0) at address L, and to reset value in p_1.

3.3 Global architecture

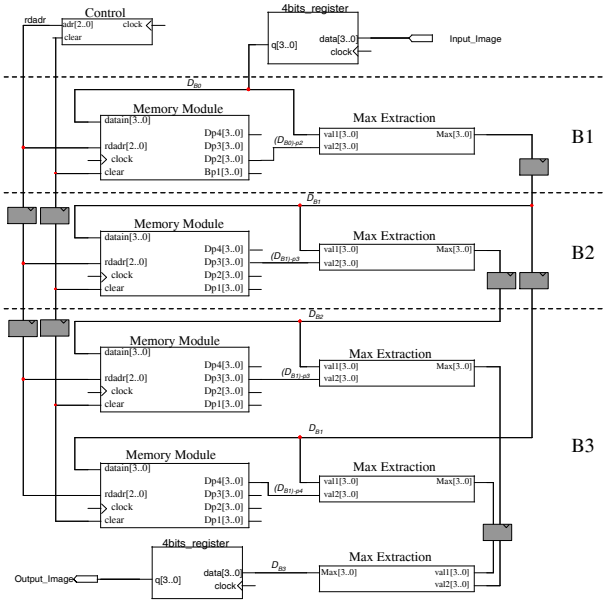

Fig. 8 Global architecture

The global architecture (see fig. 8) is composed of a simple global control (counter) providing the address and the set of pipelined elementary stage of dilation. The configuration of each stage consists of determining the number of elementary dilations and connecting the output of each memory module according to the B_i definition. The "Max extraction module" designates an elementary subtractor whose carry-out output drives the selection input of a multiplexer. The temporal performances of the system are determined by the maximal data path time fixing the minimal period. Thus the maximal frequency is dependant on $(N_D + N_U)$. A conventional solution relies on the introduction of additional registers between stages to reduce the critical data path. This method produces a constant latency in terms of clock pulses between the inputs and outputs but leads to a faster clock frequency, independently of the number of stages. These registers are represented by gray boxes in the global architecture diagram. In practice, registers are available in FPGA at the output of all logic modules. So additional registers for a pipeline implementation are only required for data paths with no operation between stages such as the address of the counter or the propagation of B_2 towards B_4.

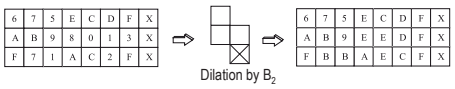

Fig. 9 Expected values for a dilation by B_2

Figure 9 shows expected results for a dilation of an input image by B_2, and figure 10 gives the results of the simulation.

3.4 Implementation

Comparative implementation efficiency with existing architectures is summarized in Table 1. Except for the Diamantaras solution, all the other techniques have been optimized for only rectangular elements. For memory and cycle requirements, the proposed method achieves the same performances as the best solutions. Only Chien architecture provides a better solution in terms of comparators, but as mentioned before, the PRR concept is essentially efficient for 1D and rectangular elements. Moreover, PRR and the proposed methods are not incompatible : additional improvements in order to reduce the number of operations, for simple shapes, can be a posteriori realized after the decomposition definition.

The proposed architecture has been synthesized into FPGAs. As the architecture requires few resources, a medium FPGA such as an APEX 20KC (Altera) used in these experiments is sufficient to support dilations by large structuring elements. Maximum clock rate is dependant only on one combinatorial stage and thus dependant only on M. For $M = 8$ the maximum clock frequency is 50 MHz. For instance, a 512×512 grayscale image is computed during one pass in 5 ms whatever the structuring element shape and size.

4 Conclusion

In this paper, we have presented an algorithm enabling the design of fast mathematical grayscale morphology operations. We have proposed a systematic way to describe any 8-convex structuring elements through a generic decomposition process using 2-pixel structuring elements. Then these 8-convex structuring elements are recursively constructed by elementary dilations and union sets, based on a fixed and close neighborhood. This decomposition system has been used to define the efficient implementation of elementary morphological operators in both software and hardware. Computation complexity is directly correlated to the number of elementary dilations involved in the decomposition.

References

Boltyanskii VG, Soltan PS (1978) Combinatorial geometry and convexity classes. Russian Mathematical Surveys 33(1):1–45

Chen S, Haralick R (1995) Recursive erosion, dilation, opening, and closing transforms. IEEE Trans on Image Processing 4(3):335–345

Chien S, Ma S, Chen L (2005) Partial-result-reuse architecture and its design technique for morphological operations with flat structuring elements. IEEE Trans on Circuits and Systems for Video Technology 15(9):1156–1169

Coltuc D, Pitas I (1998) Fast computation of a class of running filters. IEEE Trans Signal Process 46(6):549–553

Diamantaras K, Kung S (1997) A linear systolic array for real-time morphological image processing. Journal on VLSI Signal Processing 17:43–55

Eckhardt U (2001) Digital lines and digital convexity. Digital and Image Geometry 2243:209–228

Gil J, Kimmel R (2002) Efficient dilation, erosion, opening, and closing algorithms. IEEE Pattern Analysis and Machine Intelligence 24(12):1607–1617

Ji L, Piper J, Tang J (1989) Erosion and dilation of binary images by arbitrary structuring elements using interval coding. Pattern Recog Letters (9):201–249

Normand N (2003) Convex structuring element decomposition for single scan binary mathematical morphology. In: Conf. on Discrete Geometry for Computer Imagery, Naples

Ong S, Sunwoo M (1997) A new cost-effective morphological filter chip. In: IEEE Workshop Design Signal Processing Systems, pp 421–430

Pitas I (1989) Fast aglorithms for running ordering and max/min calculation. IEEE TransCircuits and Systems 36(6):795–804

Ruetz P, Brodersen R (1987) Architectures and design techniques for real-time image processing ics. IEEE Journal on Solid State Circuit 22(2):233–250

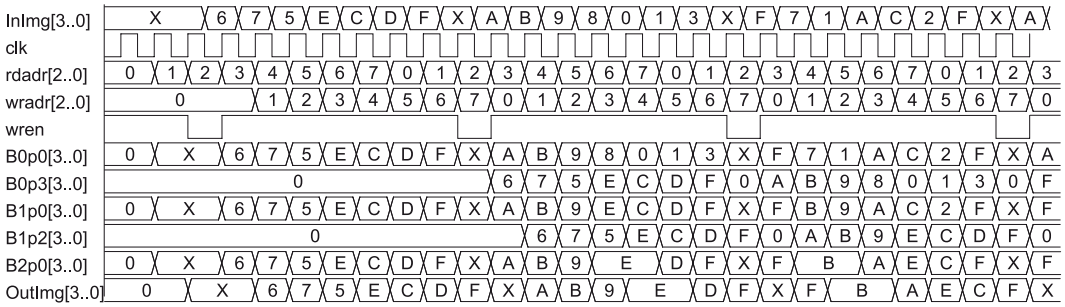

Fig. 10 Simulation results for a dilation by B_2

Serra J (1982) Image analysis and mathematical morphology. Academic Press, London

Sheu M, Wang J, Chen J, Suen A, Jeang Y, Lee J (1992) A data-reuse architecture for gray-scale morphologic operations. IEEE TransCircuits on Syst II, Analog Digital Signal Process 39(10):753–756

Soille P, Breen EJ, Jones R (1996) Recursive implementation of erosions and dilations along discrete lines at arbitrary angles 18(5):562–567

Van Herk M (1992) A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels 13(7):517–521

Wang X, Bertrand G (1988) An algorithm for a generalized distance transformation based on minkowski operations. In: 9th Int. Conf. Pattern Recognition, pp 1164–1168

Xu J (1991) Decomposition of convex polygonal morphological structuring elements into neighborhoods subsets. IEEE Trans Pattern Anal Machine Intell 13(2):153–162